Praktikum 4
PHP OOP dan MVC

Tujuan Praktikum :

- Memberikan pemahaman kepada mahasiswa tentang konsep Object Oriented Programming.
- Memberikan pemahaman kepada mahasiswa tentang implementasi konsep OOP pada bahasa scripting PHP
- Memberikan pemahaman kepada mahasiswa tentang MVC (Model View Controller) Design Pattern
- Memberikan pemahaman kepada mahasiswa implementasi MVC dengan menggunakan PHP scripting

I. Dasar Teori OOP

a. Class

Class adalah sebuah "blueprints" untuk sebuah object, dengan kata lain bahwa class adalah cetakan dari object. Pada bahasa pemrograman, class merupakan sekelompok kode yang dituliskan untuk mendefinisikan properti dan metod yang ada pada sebuah object. Berikut adalah contoh script PHP untuk membuat class.

```
1   <?php
2         Class Demo
3         {
4
5         }
6   ?>
```

Class didefinisikan dengan memuat properti dan metode, dimana properti adalah sebuah data yang menjelaskan tentang class dan metode adalah tingkah laku yang bisa dilakukan oleh object. Berikut adalah contoh kode sebuah class yang dilengkapi dengan properti dan metode.

```
1   <?php
2         Class Demo
3         {
4   $nama; // properti
5
6               function sayHallo() //metode
7               {
8                     echo "Hallo $this->nama";
9               }
10        }
11  ?>
```

b. Object

Object adalah hasil instansiasi dari class, dan mengandung seluruh *resource* yang telah didefinisikan pada class. Berikut adalah cara meng-instansiasi object dari class yang sudah didefinisikan.

```
1   <?php
2
3   require_once('file-class.php'); //file yang memuat class
4
5   $obj = new Demo(); //proses instansiasi object
6
```

```
7   ?>
```

Karena class merupakan "cetakan" atau "blueprints" dari object, maka object hasil instansiasi juga mempunyai *resource* seperti class. Berikut contoh kode memanggil properti dan metode.

```
1   <?php
2
3   require_once('file-class.php'); //file yang memuat class
4
5   $obj = new Demo(); //proses instansiasi object
6
7   $obj->nama = "Achmad Widhy"; //mendefinisikan nilai properti
8   echo $ojb->nama; // memanggil properti
9   $obj->sayHallo(); // menjalankan metode
10  ?>
```

Maka apabila dijalankan akan menghasilkan :

Hallo Achmad Widhy

c. Encapsulation

Encapsulation adalah mekanisme "membungkus" sebuah data pada sebuah object. Dalam istilah lain yang sertingkali dilontarkan adalah "Information Hiding". Pada PHP terdapat 3 modifier yang dapat diimplementasikan untuk melakukan "pembungkusan" data yaitu *private*, *protected* dan *public*. Modifier tersebut digunakan untuk mendefinisikan tingkat visibilitas sebuah data (properti) atau fungsi (metode) yang ada di dalam class.

Modifier	Keterangan
Public	Untuk mendefinisikan data atau metode yang akan terlihat dari luar oleh siapapun dan dimanapun.
Private	Untuk mendefinisikan data atau metode agar hanya terlihat pada class/object itu sendiri.
Protected	Untuk mendefinisikan data atau metode untuk tidak terlihat dari luar (seperti private), tetapi akan dapat diakses oleh "anak" dari class tersebut.

Berikut adalah contoh kode implementasi dari encapsulasi.

```
1   <?php
2   class Person
3   {
4       private $_name;
5       private $_age;
6
7       function __construct($name, $age = 0)
8       {
9           if (!is_int($age))
10          {
11              throw new Exception("Cannot assign non integer value to
12  integer field, 'Age'");
13          }
14
15          $this->_age = $age;
16          $this->_name = $name;
17      }
18
```

```
19      public function setAge($age)
20       {
21           if (!is_int($age))
22           {
23               throw new Exception("Cannot assign non integer value to
24  integer field, 'Age'");
25           }
26
27           $this->_age = $age;
28       }
29
30      public function yearsToRetire()
31       {
32           return 67 - $this->_age;
33       }
34  }
35
36  $person = new Person("Wes");
37  $person->setAge(31);
38
39  echo $person->yearsToRetire();
40
41  ?>
```

Properti _name dan _age mempunyai modifier private, dimana tidak akan bisa dikases langsung. Properti tersebut akan bisa diakses jika kita mendefinisikan sebuah metode untuk mengakses properti tersebut yang memiliki modifier public.

d. Constructor dan Destructor

PHP memungkinkan pengembanguntuk menyatakanmetodekonstruktor untuk sebuahclass.Classyangmemiliki metodekonstruktormemanggil metode inipada setiap objekyang baru dibentuk (diinstansiasi), sehingga sangat cocokuntukinisialisasibahwa objekmungkin perlusebelum digunakan. Berikut contoh kode konstruktor.

```
1   <?php
2
3   class Person {
4      function __construct() {
5          print "Konstruktor Person\n";
6      }
7   }
8
9   $obj = new Person();
10
11  ?>
```

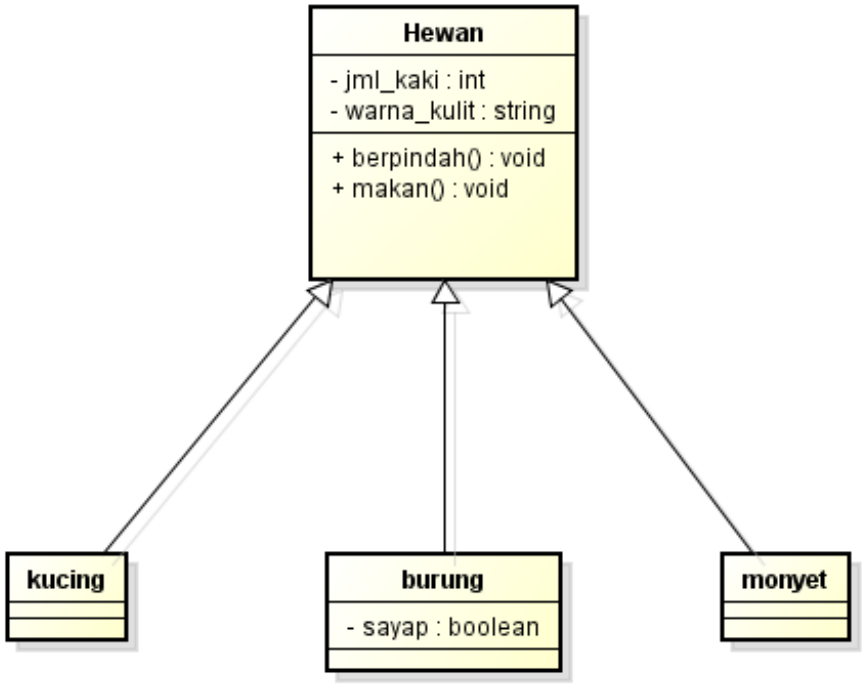
PHP memperkenalkankonsepdestructorsama dengan yanglainbahasa berorientasi objek, seperti C++.Metodedestruktorakan dipanggilsegera setelahtidak ada referensilain untukobjek tertentu. Berikut contoh kode destruktor.

```
1   <?php
2
3   class Person {
4      function __destruct() {
5          print "Destruktor Person\n";
6      }
7   }
8
9   $obj = new Person();
```

```
10
11  ?>
```

e. Inheritance

Dalam pemrograman berorientasi objek(OOP), *Inheritance* (pewarisan)adalah cara untuk menggunakan kembali kode objek yang ada, atau untuk mendirikan subtype dari objek yang sudah ada, atau keduanya, tergantung pada dukungan bahasa pemrograman. Dalam warisanklasik dimana objek yang didefinisikan oleh class, class dapat mewarisi atribut dan perilaku dari class yang sudah ada yang disebut class atau super classes atau class induk atau class leluhur. Class baru yang dikenal sebagai class turunan atau subclass atau class anak. Hubungan class melalui warisan menimbulkan hirarki. Berikut contoh kasus inheritance.



Kode dari class diagram diatas adalah sebagai berikut.

```
1   <?php
2          //Class hewan
3   class hewan
4          {
5                 protected $jml_kaki;
6                 protected $warna_kulit;
7
8                 function __construct()
9                 {
10                }
11
12                function berpindah()
13                {
14                       echo "Saya berpindah";
15                }
16
17                function makan()
18                {
19                       echo "Saya makan";
20                }
21         }
22
23
```

```
24          //Class kucing
25  class kucing extends hewan
26          {
27                  function berpindah()
28                  {
29                          echo "Saya merangkak dengan 4 kaki";
30                  }
31          }
32
33          //Class burung
34  class burung extends hewan
35          {
36                  protected $sayap;
37
38                  function berpindah()
39                  {
40                          echo "Saya terbang";
41                  }
42
43                  function makan()
44                  {
45                          echo "Saya makan dengan mematuk";
46                  }
47
48          }
49
50          //Class monyet
51  class monyet extends hewan
52          {
53
54          }
55
56  ?>
```

f. Final Keyword

PHP memperkenalkan "final" keyword, dimana ini akan mencegah proses overriding method pada class anak (sub-class). Hal ini bisa kita terapkan pada metode dan class. Apabila metode kita berikan status final, maka metode tersebut tidak akan bisa dioverride, begitu juga pada class, apabila kita berikan status "final" pada deklarasi class maka class tersebut tidak bisa diperpanjang (diwariskan). Contoh kode menggunakan "final" keyword.

- Final keyword pada metode

```
1   <?php
2   class BaseClass {
3      public function test() {
4          echo "BaseClass::test() called\n";
5      }
6
7      final public function moreTesting() {
8          echo "BaseClass::moreTesting() called\n";
9      }
10  }
11
12  class ChildClass extends BaseClass {
13     public function moreTesting() {
14         echo "ChildClass::moreTesting() called\n";
15     }
16  }
17
18  // Results in Fatal error: Cannot override final method BaseClass::moreT
19  esting()
20  ?>
```

```
21
```

- Final keyword pada class

```
1   <?php
2   final class BaseClass {
3      public function test() {
4          echo "BaseClass::test() called\n";
5      }
6
7      // Here it doesn't matter if you specify the function as final or not
8      final public function moreTesting() {
9          echo "BaseClass::moreTesting() called\n";
10     }
11  }
12
13  class ChildClass extends BaseClass {
14  }
15  // Results in Fatal error: Class ChildClass may not inherit from final c
16  lass (BaseClass)
17  ?>
```

g. Class Abstraction

PHP memperkenalkan abstract class dan abstract method. Class yang didefinisikan sebagai abstract tidak bisa diinstasiasi, dan class yang terdiri paling tidak satu method abstract harus didefinisikan sebagai abstract class. Class abstract hanya bisa mewariskan resources nya pada class anaknya. Setiap class yang mewarisi class abstract, wajib menuliskan seluruh method abstract yang dipunyai oleh parent class-nya (super class). Berikut contoh pendeklarasian class abstract.

```
1   <?php
2   abstract class AbstractClass
3   {
4       // Force Extending class to define this method
5       abstract protected function getValue();
6       abstract protected function prefixValue($prefix);
7
8       // Common method
9       public function printOut() {
10          print $this->getValue() . "\n";
11      }
12  }
13
14  class ConcreteClass1 extends AbstractClass
15  {
16      protected function getValue() {
17          return "ConcreteClass1";
18      }
19
20      public function prefixValue($prefix) {
21          return "{$prefix}ConcreteClass1";
22      }
23  }
24
25  class ConcreteClass2 extends AbstractClass
26  {
27      public function getValue() {
28          return "ConcreteClass2";
29      }
30
31      public function prefixValue($prefix) {
32          return "{$prefix}ConcreteClass2";
33      }
```

```
}

$class1 = new ConcreteClass1;
$class1->printOut();
echo $class1->prefixValue('FOO_') ."\n";

$class2 = new ConcreteClass2;
$class2->printOut();
echo $class2->prefixValue('FOO_') ."\n";
?>
```

h. Object Interfaces

Object Interface memungkinkan kita untuk *membuat kode yang menentukan method mana yang akan diimplementasikan tanpa harus mendefinisikan bagaimana method tersebut akan bekerja (hanya nama method saja)*. Interface didefinisikan dengan "Interface" keyword, mirip dengan deklarasi class biasa, hanya saja definisi atau detail method tidak dituliskan. Seluruh method yang dideklarasikan pada interface harus memiliki modifier "public". Untuk mengimplementasikan sebuah interface, kita dapat menggunakan "implement" keyword. Seluruh method yang ada pada interface harus diimplementasikan seluruhnya. Sebuah class bisa mengimplementasikan lebih dari satu interface.

Catatan :

- Class tidak bisa mengimplementasikan dua interface yang mempunyai nama method yang sama.
- Interface bisa diwariskan seperti class menggunakan "extends".
- Class yang mengimplementasikan interface harus menggunakan method-method yang ada pada interface tersebut dengan nama dan spesifikasi yang sama persis.

Berikut adalah contoh kode tentang object interfaces.

```
<?php
interface a
{
    public function foo();
}

interface b extends a
{
    public function baz(Baz $baz);
}

// This will work
class c implements b
{
    public function foo()
    {
    }

    public function baz(Baz $baz)
    {
    }
}

// This will not work and result in a fatal error
class d implements b
{
    public function foo()
    {
    }

    public function baz(Foo $foo)
    {
    }
```

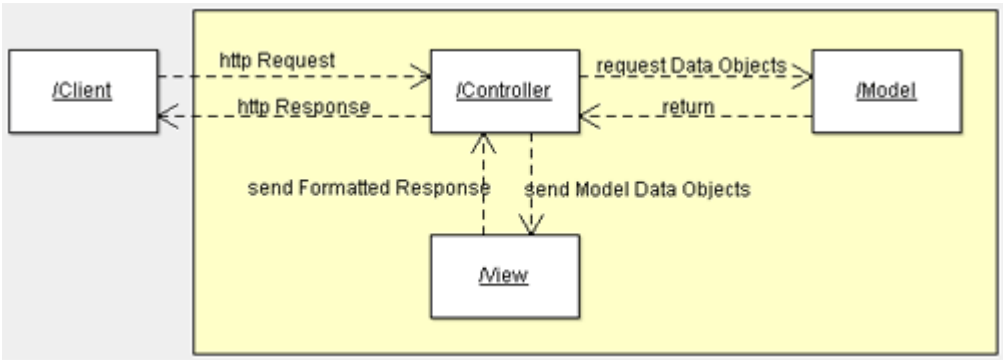
```
34  }
35  ?>
```

II. Dasar Teori MVC (Model View Controller)

MVC (Model View Controller) pattern adalah sebuah pattern yang banyak digunakan untuk membangun aplikasi web saat ini. MVC pattern terbagi menjadi 3 modul, Model, View dan Controller :

- Model, berfungsi untuk mengontrol data, disini dilakukan proses pengambilan dan penyimpanan data sebuah sistem, biasanya data berasal dari database yang digunakan oleh sistem.
- View, bertanggung jawab untuk mengatur tampilan dengan format yang spesifik.
- Controller, bertanggung jawab untuk meng-handle model dan view layer untuk digabungkan menjadi satu. Controller berposisi di tengah, menghubungkan model dan view, dan sebagai tujuan utama user dalam melakukan request.

Berikut adalah gambaran MVC pattern :



Gambar 1. MVC Pattern

Contoh sederhana penerapan MVC pada aplikasi Web dengan PHP, pertama yang kita lakukan adalah mendefinisikan model.

1. Model
   Model adalah layer yang bertanggung jawab untuk melakukan hubungan dengan database, untuk contoh kali ini tidak menggunakan database, asumsi kita adalah layer model telah berhasil mendapatkan data dari database. Untuk contoh kali ini kita mengambil topik tentang database buku. Karena kita menggunakan topik buku maka untuk tahap awal kita akan menuliskan kode untuk class book.

   File book.php

```
1   <?
2   class book {
3
4   public $judul;
5   public $pengarang;
6   public $penerbit;
7   public $tahun;
8
9   public function __construct($title,$author,$publisher,$year)
10  {
11          $this->judul = $title;
```

```
12          $this->pengarang = $author;
13          $this->penerbit = $publisher;
14          $this->tahun = $year;
15  }
16
17  }
18
19  ?>
```

Class book, adalah class untuk merepresentasikan buku.

File model.php

```
1   <?
2   include_once'book.php';
3
4   class model {
5
6           public function getData()
7           {
8           return array(
9                           new book('Pemrograman PHP & MySql','Bayu
10  Priyambadha','UB Press','2011'),
11                          new book('Pemrograman Framework MVC dengan
12  PHP','Widhy','UB Press','2011'),
13                          new book('Membangun Aplikasi Web dengan PHP dan
14  AJAX','Achmad Arwan','UB Press','2012'),
15                          new book('Kolaborasi PHP, MVC dan AJAX','Bayu
16  Priyambadha','UB Press','2012')
17                          );
18          }
19  }
20
21  ?>
```

Model mempunyai method getData() yang menghasilkan output berupa array dengan isi adalah objek book.

2. View
   Untuk layer view, kita hanya akan mendefinisikan sebuah template html sebagai tempat untuk menampilkan data. Berikut adalah skrip html untuk layer view.

File view.php

```
1   <html>
2           <head>
3                   <title>MVC dengan PHP</title>
4           </head>
5           <body>
6                   <center>
7
8                           <?=$data?>
9
10                  </center>
11          </body>
12  </html>
```

Kode php yang ada diantara tag html diatas berfungsi untuk menampilkan data yang dikirimkan oleh controller.

3. Controller

Sebagai layer yang berfungsi sebagai "play maker", controller harus mempunyai akses ke model dan view. Berikut adalah kode untuk controller.

File controller.php

```
<?
include_once'model.php';

class controller {

	function invoke()
	{
		$model_data = new model();
		$row_data = $model_data->getData();

		$data ="<table border=1>

	<tr><th>JUDUL</th><th>PENGARANG</th><th>PENERBIT</th><th>TAHUN</th
></tr>";
		foreach ($row_data as $key => $value) {
			$data .="<tr><td>".$value->judul."</td><td>".$value-
>pengarang."</td><td>".$value->penerbit."</td><td>".$value-
>tahun."</td></tr>";
		}
		$data .="</table>";
		include('view.php');
	}
}

?>
```

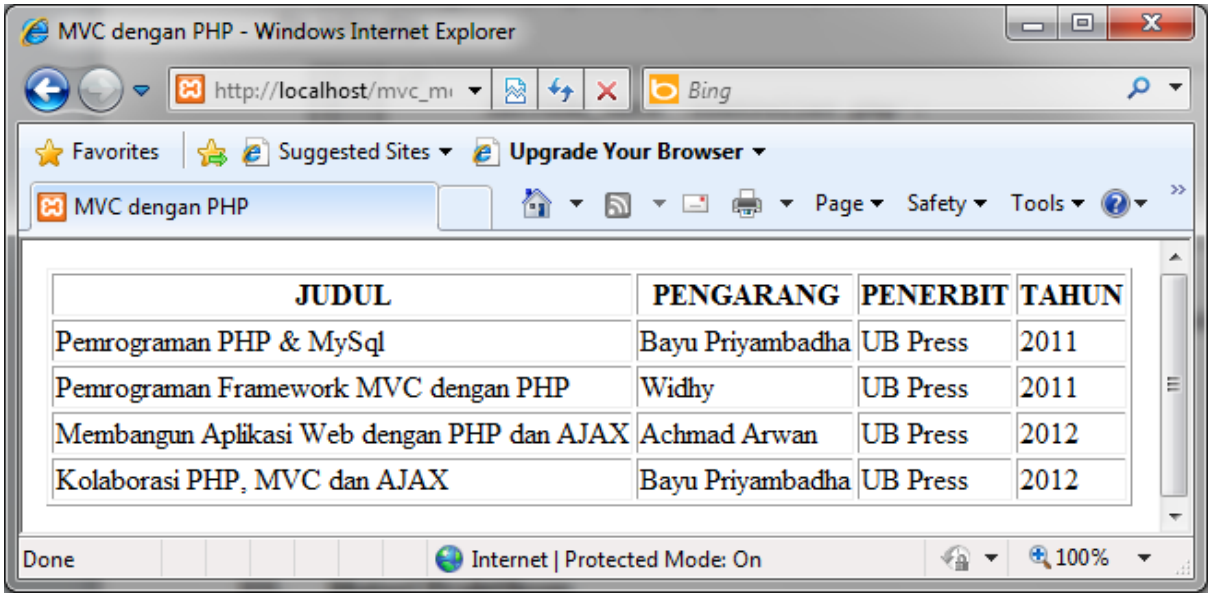
Controller mempunyai method invoke, dimana di method tersebutlah proses penyatuan data dari model dan view digabungkan.

Setelah ketiga layer selesai dibuat, maka tahap terakhir adalah membuat file index.php, dimana file ini adalah sebagai file penghubung yang diakses pertama kali user melakukan request. Berikut adalah file index.php.

```
<?
	include_once'controller.php';

	$main_ctrl = new controller();
	$main_ctrl->invoke();
?>
```

Pada file index.php ini, controller diinstansiasi, dan method invoke dijalankan. Apabila berhasil akan menghasilkan seperti berikut :



JUDUL	PENGARANG	PENERBIT	TAHUN
Pemrograman PHP & MySql	Bayu Priyambadha	UB Press	2011
Pemrograman Framework MVC dengan PHP	Widhy	UB Press	2011
Membangun Aplikasi Web dengan PHP dan AJAX	Achmad Arwan	UB Press	2012
Kolaborasi PHP, MVC dan AJAX	Bayu Priyambadha	UB Press	2012

Gambar 2. Tampilan Hasil

III. Materi Praktikum

Latihan 1: Class dan Object + Modifier

Tuliskan code dibawah ini, simpan dalam file Lat4_1.php!

```
1   <?php
2   //class mobil
3   Class Mobil{
4           public $nama;
5           public $merk;
6
7           function getInfo(){
8                   echo "Nama mobil: ".$this->nama."<br/>";
9                   echo "Merk: ".$this->merk."<br/>";
10          }
11  }
12
13  //bagian main
14  $ferari=new Mobil();
15  $ferari->nama="xxx";
16  $ferari->merk="aaa";
17
18  $ferari->getInfo();
19  ?>
```

a. Bagaimanahasiltampilan di atas?
b. Buatlahsebuah method overload getInfodengan parameter $a. Lalujalankandanamatiperubahan yang terjadi.
c. Lalu simpulkan apa yang Anda peroleh dari Latihan 1!

Latihan 2:

Tuliskan code dibawah ini, simpan dalam file Lat4_2a.php!

```
1   <?php
2   Class mahasiswa{
3           public $nama;
4           public $nim;
5
6           function __construct($a,$b){
```

```
7               $this->nama=$a;
8               $this->nim=$b;
9               echo "Kelas telah dibuat<br/><br/>";
10              }
11
12      function cetak(){
13              echo $this->nama."<br/>".$this->nim."<br/><br/>";
14              }
15
16      function __destruct(){
17              echo "Kelas telah dihancurkan";
18              }
19  }
20
21  ?>
```

Tuliskan code dibawah ini, simpan dalam file Lat4_2b.php!

```
1   <?php
2
3   require_once("lat4_2a.php");
4
5   $mhs2=new mahasiswa("Pennyka","0910683073");
6
7   $mhs2->cetak();
8   ?>
```

Modifikasilah latihan 4_2a dengan menambahkan 1 konstruktor lagi di lat 4_2a baris 11. Lalu jalankan. Bagaimana hasil tampilan di atas sebelum dan sesudah dimodifikasi? Lalu simpulkan apa yang Anda peroleh dari Latihan 2!

Latihan 3:

Tuliskan code dibawah ini, simpan dalam file Lat4_3a.php!

```
1   <?php
2
3   class mahasiswa{
4           private $nama;
5           private $nim;
6
7           function __construct(){}
8
9           function setNama($a){
10              $this->nama=$a;
11              }
12
13          function setNim($b){
14              $this->nim=$b;
15              }
16
17          function getNama(){
18              return $this->nama;
19              }
20
21          function getNim(){
22              return $this->nim;
23              }
24
25          function destruct(){}
26  }
27
28  ?>
```

Tuliskan code dibawah ini, simpan dalam file Lat4_3b.php!

```
1   <?php
2
```

```
require_once("lat4_3a.php");

$mhs1=new mahasiswa();
$mhs1->nama="hendra";
echo $mhs1->nama;

?>
```

a. Apakah program error? Jika error mengapahalitudapatterjadi?
b. Rubahlah modifier dari variable namadannimmenjadi protected dan public. Laluamatiperubahan yang terjadi.
c. Modifikasilah Lat4_3b sehingga dapat member dan mencetak isi dari nim dan nama dengan modifier private
d. Simpulkanapa yang andaperolehdarilatihan 3!

Jawab:

Latihan 4:

Tuliskan code dibawah ini, simpan dalam file Lat4_4a.php!

```
<?php

require_once("lat4_3a.php");

class asisten extends mahasiswa
{
       function __construct(){}
}
```

Tuliskan code dibawah ini, simpan dalam file Lat4_4b.php!

```
<?php

require_once("lat4_4a.php");

$as = new asisten();
$as->setNama("tes");
echo $as->getNama();
```

Simpulkan apa yang Anda peroleh dari Latihan 4!

Latihan 5: Abstract Class

Tuliskan code dibawah ini, simpan dalam file Lat4_5a.php!

```
<?php

abstract class mahasiswa
{
    abstract protected function getTugasAkhir();
    abstract protected function getProgram($postfix);

    public function tugasAkhir()
    {
        print $this->getTugasAkhir() . "<br>";
    }
}

class sarjana extends mahasiswa
{
    protected function getTugasAkhir()
    {
        return "Skripsi";
    }

    public function getProgram($postfix)
```

```
22      {
23          print "{$postfix} S1";
24      }
25  }
26
27  class magister extends mahasiswa
28  {
29      public function getTugasAkhir()
30      {
31          return "Tesis";
32      }
33
34      public function getProgram($postfix)
35      {
36          print "{$postfix} S2";
37      }
38  }
```

Tuliskan code dibawah ini, simpan dalam file Lat4_5b.php!

```
1   <?php
2
3   require_once("lat4_5a.php");
4
5   $s = new sarjana;
6   $s->getProgram('Mahasiswa') ."<br>";
7   $s->tugasAkhir();
8
9   $m = new magister;
10  $m->getProgram('Mahasiswa') ."<br>";
11  $m->tugasAkhir();
```

a. Bagaimana hasil tampilan dari program di atas?
b. Hapuslah kode baris 29 – 32 pada lat4_5a.php, lalu jalankan lat4_5b.php. Bagaimana hasil tampilan program di atas? Jelaskan mengapa hal tersebut terjadi?
c. Simpulkan apa yang anda peroleh pada latihan 4?

Latihan 6:

Tuliskankodeberikutpada lat4_6.php

```
1   <?php
2   interface a
3   {
4       public function foo();
5   }
6
7   interface b
8   {
9       public function bar();
10  }
11
12  interface c extends a, b
13  {
14      public function baz();
15  }
16
17  class d implements c
18  {
19      public function foo()
20      {
21      }
22
23      public function bar()
24      {
```

```
25      }
26
27      public function baz()
28      {
29      }
30  }
31  ?>
```

a. Jelaskan maksud dari program di atas?
b. Hapuslah kode baris 27 – 29, lalu jalankan lat4_6.php. Bagaimana tampilan program di atas? Jelaskan mengapa hal tersebut dapat terjadi?
c. Dari contoh kode diatas, buatlah class baru dengan nama "e" yang mempunyai method **foo** dan **bar**.
d. Simpulkan apa yang anda peroleh darilatihan 6!

Latihan 7:

Tuliskan kode berikut pada lat4_7.php

```
1   <?php
2   class A
3
4   {
5
6   final public function disp(){
7
8   echo "Inside the final function";
9
10  }
11
12  }
13
14  class B extends A{
15
16  function disp(){
17
18  echo "Inside the final function";
19
20  }
21
22  }
23
24  $obj=new B();
25
26  $obj->disp();
27
    ?>
```

a. Bagaimana tampilan program di atas?Jelaskan mengapa hal tersebut terjadi?
b. Modifikasi program di atas dengan menghapus kata final pada kode baris 5 dan menambahkan kata final pada baris 2. Bagaimana tampilan program di atas? Jelaskan mengapa hal tersebut terjadi?
c. Simpulkan apa yang anda peroleh dari latihan 7!

Latihan 8:

Tuliskan kode berikut pada lat4_8.php

```
1   <?php
2
```

```
3   class One{
4
5   private static $var=0;
6
7                   function __construct(){}
8
9                   static function disp(){
10
11                          print self::$var;
12                  }
13
14                  function __destruct(){}
15  }
16
17  One::disp();
18
19  ?>
```

Simpulkan apa yang anda peroleh dari latihan 8!

Latihan 9:

Dari contoh kode MVC pada penjelasan diatas buatlah :

1. Model berhubungan langsung dengan database mysql dengan sepesifikasi sebagai berikut :
   a. Database : Library
   b. Table : book

   Username dan password menyesuaikan.

2. Buatlah view menjadi sebuah object, susun class view dimana fungsinya adalah mengambil file-file template yang sudah disediakan!
3. Tuliskan masing-masing kodenya, buatlah simulasinya!